This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Previously Presented) A method of distributing an object-oriented computer program comprising:

(a) translating the program bytecode into machine independent virtual processor code which uses an instruction set of a virtual processor;

(b) transmitting the virtual processor code from a server to a client device; and

(c) translating the virtual processor code into native code which uses an instruction set of a physical processor of the client device,

wherein the bytecode is stack-based, and in which the virtual processor code is register-based.

2. (Previously Presented) The method as claimed in claim 1 in which the program bytecode includes a class file, the class file being converted into one or more virtual processor tools which use the instruction set of the virtual processor.

3. (Previously Presented) The method as claimed in claim 2 in which the class file includes a plurality of methods, and which some or all the methods in the class file are converted to a respective virtual processor tool.

4. (Previously Presented) The as claimed in claim 2 in which the class file includes a call to a method, and in which the virtual processor code provides a call to a corresponding tool.

5. (Previously Presented) The method as claimed in claim 2 in which the class file includes a reference to a field, and in which the virtual processor code provides a fixup tool for use in locating the field.

6. (Previously Presented) The method as claimed in claim 5 in which the fixup tool is arranged to return a constant fixup value which is representative of the offset of the said field within an object.

7. (Previously Presented) The method as claimed in claim 6 including linking the virtual processor code and determining the constant fixup value in dependence upon virtual processor code which has been translated from another class file.

8. (Previously Presented) The method as claimed in claim 6 in which the fixup tool returns a value which is used to patch a method which gets or puts the value of a field.

9. (Previously Presented) The method as claimed in claim 2 in which the virtual processor code has, included within it at a plurality of points, fixup instructions which indicate that the code at the said points has to be modified by the respective fixup instruction prior to use.

10. (Previously Presented) The method as claimed in claim 7 in which the fixup instructions provide instructions as to how the native code can reference another class, or a field or method in another class.

11. (Previously Presented) The method as claimed in claim 9 in which the fixup instructions are transferred, functionally unaltered, by the native translator into the native code; the fixup instructions being replaced with native instructions when the native code is bound on the said real physical processor.

12. (Canceled)

13. (Previously Presented) A method of executing an object oriented computer program comprising:

translating the program bytecode into machine independent virtual processor code which uses an instruction set of a virtual processor;

transmitting the virtual processor code from a server to a client device; and

translating the virtual processor code into native code which uses an instruction set of a physical processor of the client device; and

executing the native code on the physical processor,

wherein the byte code is stack-based and the virtual processor code is register-based.

14. (Previously Presented) The method as claimed in claim 13 including binding the translated tools into a task, and executing the task in native code on the physical processor.

15. (Currently Amended) A computer system ~~adapted to carry out the method as claimed in claim 1~~ comprising one or more processors for implementing control over the following method:

(a) translating the program bytecode into machine independent virtual processor code which uses an instruction set of a virtual processor;

(b) transmitting the virtual processor code from a server to a client device; and

(c) translating the virtual processor code into native code which uses an instruction set of a physical processor of the client device,

wherein the bytecode is stack-based, and in which the virtual processor code is register-based.

16. (Previously Presented) The method as claimed in claim 1 which includes:

(d) transmitting the virtual processor code from the server to a second client device; and

(e) translating the virtual processor code into different native code which uses an instruction set of a physical processor of the second client device.

17. (Previously Presented) The method as claimed in claim 13 including executing the different native code on the physical processors of different client devices.

18. (Currently Amended) A computer system ~~adapted to carry out the method as claimed in claim 16.~~ according to claim 15, wherein the method further includes

(d) transmitting the virtual processor code from the server to a second client device; and

(e) translating the virtual processor code into different native code which uses an instruction set of a physical processor of the second client device.

19. (Previously Presented) A distributed computer system comprising a server including a store for storing virtual processor code, said code being a machine-independent representation of the bytecode of an object oriented computer program using an instruction set of a virtual processor, and a plurality of remote client devices in communication with the server, each client device including a client processor, a native translator arranged to translate the virtual processor code into native code which uses the instruction set of the respective client processor, and a native code store; the system including transmission means for transmitting the virtual processor code from the server to the client devices,

wherein the bytecode is stack-based, and in which the virtual processor code is register-based.

20. (Previously Presented) The distributed computer system as claimed in claim 19 in which the transmission means consists of or includes a wireless network.

21. (Previously Presented) The distributed computer system as claimed in claim 20 in which the client devices are mobile phones.

22. (Previously Presented) The distributed computer system as claimed in claim 20 in which the client devices are hand-held computers.

23. (Previously Presented) The distributed computer system as claimed in claim 19 in which the client devices are hand-held games consoles.

24. (Previously Presented) The distributed computer system as claimed in claim 19 in which at least one of the client devices includes a first type of client processor and in which at least another of the client devices includes a second type of client processor, using a different instruction set from that of the first type.

25. (Previously Presented) The distributed computer system as claimed in claim 19 in which the server is further arranged to translate the object-oriented computer program from bytecode into virtual processor code.

26. (Previously Presented) The method as claimed in claim 2, including verifying the integrity of the class bytecode, and of any external calls.

27. (Previously Presented) The method as claimed in claim 2, in which the class file is a Java class file.

28. (Previously Presented) The method as claimed in claim 2, in which the step of translating the program bytecode into virtual processor code is carried out by a first translator program which is itself written in virtual processor code.

29. (Previously Presented) The method as claimed in claim 1, in which the step of translating the virtual processor code into native code is carried out by a second translator program which is itself written in virtual processor code.

30. (Currently Amended) A computer-readable medium encoded with a computer program which when executed by a computer performs the following method: ~~for performing the method as claimed in claim 1.~~

(a) translating the program bytecode into machine independent virtual processor code which uses an instruction set of a virtual processor;

(b) transmitting the virtual processor code from a server to a client device; and

(c) translating the virtual processor code into native code which uses an instruction set of a physical processor of the client device,

wherein the bytecode is stack-based, and in which the virtual processor code is register-based.

31. (Canceled)

32. (Canceled)